Energy-Efficient Task Offloading and Resource Allocation via Deep Reinforcement Learning for Augmented Reality in Mobile Edge Networks

Xing Chen and Guizhong Liu, *Member, IEEE*

***Abstract*—The augmented reality (AR) applications have been widely used in the field of Internet of Things (IoT) because of good immersion experience for users, but their ultralow delay demand and high energy consumption bring a huge challenge to the current communication system and terminal power. The emergence of mobile-edge computing (MEC) provides a good thinking to solve this challenge. In this article, we study an energy-efficient task offloading and resource allocation scheme for AR in both the single-MEC and multi-MEC systems. First, a more specific and detailed AR application model is established as a directed acyclic graph according to its internal functionality. Second, based on this AR model, a joint optimization problem of task offloading and resource allocation is formulated to minimize the energy consumption of each user subject to the latency requirement and the limited resources. The problem is a mixed multiuser competition and cooperation problem, which involves the task offloading decision, uplink/downlink transmission resources allocation, and computing resources allocation of users and MEC server. Since it is an NP-hard problem and the communication environment is dynamic, it is difficult for genetic algorithms or heuristic algorithms to solve. Therefore, we propose an intelligent and efficient resource allocation and task offloading algorithm based on the deep reinforcement learning framework of multiagent deep deterministic policy gradient (MADDPG) in a dynamic communication environment. Finally, simulation results show that the proposed algorithm can greatly reduce the energy consumption of each user terminal.**

***Index Terms*—Augmented reality (AR), deep reinforcement learning, Internet of Things (IoT), mobile-edge computing (MEC), multiagent deep deterministic policy gradient (MADDPG), resource allocation, task offloading.**

I. Introduction

With the full development of the fifth-generation mobile communication research, a large number of new video applications are emerging, such as augmented reality (AR) and virtual reality (VR). These new video applications can bring a better experience due to the characteristics of immersive scenes. Therefore, it is applied in many fields, such as the Internet of Things (IoT), Education, and Telemedicine [1]–[3]. In the central cloud computing system, the task with a large computing load will be offloaded to the central cloud for processing. However, the central cloud system is not enough to support these applications with large computing load and low delay requirements. The reasons are that the central cloud is far away from users and the computing load is high, so the service quality cannot be guaranteed. As a main evolutional technology in the 5G communication system, mobile-edge computing (MEC) [4], [5] provides a good direction to solve these problems because of its powerful intelligent storage and computing capabilities [6]. MEC sinks the central cloud computing service to the edge of the core network, which is closer to the data generated by users. When users request contents or offload tasks, they can access the MEC server directly instead of accessing the remote central cloud. By this way, the backhaul load, the number of served users, and the distance of transmission links can be greatly reduced. Therefore, it can provide the high bandwidth and low delay network services to users.

Based on the above analysis, how to offload tasks and which task is offloaded become a key problem. The task offloading problem in the communication system will inevitably involve the allocation of computing and transmission resources [7]. Therefore, this problem can be easily transformed into a finite resource allocation problem [8], [9]. Task modeling is an important premise to explore this optimization problem. On the one hand, an application is modeled as a whole task, which is not split in most existing researches [10], [11]. On the other hand, an application is arbitrarily split into many subtasks, and there is no dependency between subtasks [9], [12]. The above task modeling methods did not conform to reality, which was easy to cause resource waste. From the perspective of Quality of Experience (QoE) and limited device power, delay and energy consumption are generally selected as the optimization objective. Tran and Pompili [13] aimed to maximize the task offloading gains, which were measured by a weighted sum of reductions in task completion time and energy consumption. The objective of [14] was to minimize the overall energy consumption at all system entities. But each user only pays attention to their QoE and energy consumption in real life.

In this article, we focus on the optimization problem of AR application offloading in both the single-MEC and multi-MEC systems, respectively. The optimization problem of a single MEC system is presented in the previous stage work [15], which is not detailed. Based on the above analysis,

Manuscript received November 8, 2020; revised December 11, 2020; accepted January 6, 2021. Date of publication January 11, 2021; date of current version June 23, 2021. This work was supported by Shaanxi Key Research and Development Program under Grant 2018ZDCXL-GY-04-03-02. *(Corresponding author: Guizhong Liu.)*

The authors are with the School of Information and Communications Engineering, Xi'an Jiaotong University, Xi'an 710049, China (e-mail: xing_chen@stu.xjtu.edu.cn; liugz@xjtu.edu.cn).

Digital Object Identifier 10.1109/JIOT.2021.3050804

an AR application is split into five parts, and there is a certain dependency between these parts. The AR application is modeled as a directed acyclic graph, which is described in Section III-A. The optimization objective is to minimize each user's energy consumption based on meeting the delay requirements. Therefore, the problem can be formulated as a joint optimization of resource allocation and task offloading, which is a mixed problem of cooperation and competition. The contributions of this article can be summarized as follows.

1) *More Detailed Task Modeling:* A more specific and detailed task model is established to reasonably utilize resources and reflect the task execution process. The AR application is split into five parts according to the actual execution process, and there is a certain dependency between the parts. During joint optimization, an AR application is modeled as a directed acyclic graph to design optimization strategy.
2) *More Realistic Optimization Objective Setting:* In real life, we only care about the energy consumption of our own terminal equipment, not other users' energy consumption. But in the existing algorithms, most of them are to minimize the sum of energy consumption of all users, which can cause some extreme situations. Some users' energy consumption is larger and some users' energy consumption is less, which cannot guarantee fairness between users. Therefore, the optimization objective of our setting is to minimize the energy consumption of each user without exceeding the delay requirements.
3) *Joint Optimization Algorithm of Resource Allocation and Task Offloading Based on Deep Reinforcement Learning in the Single-MEC System and the Multi-MEC Collaboration System:* The environment of this joint optimization is a mixed problem of multiuser competition and cooperation. Competition means that users compete for communication and computing resources. Cooperation means that each user can complete the AR application and satisfy the delay requirements. The problem cannot be solved well by general heuristic and genetic algorithms. The main reason is that these algorithms have more iterations and less stability according to the dynamic environment. Therefore, we first model the joint optimization problem as a Markov decision process (MDP) and then utilize the multiagent deep deterministic policy gradient (MADDPG) framework to solve. MADDPG can deal with the dynamic environment after training and owns better efficiency and stability. The algorithm design and experimental verification are carried out for single-MEC systems and multi-MEC systems, respectively.

The remainder of this article is organized as follows. Section II reviews the development of the related work. Section III builds the AR application, communication, caching, and computation model. Section IV formulates the mathematical problem and models the joint optimization problem as MDP. The joint optimization algorithm based on the MADDPG framework is proposed. Section V provides the simulation result and evaluates the performance of the proposed algorithm. Finally, Section VI concludes this article.

II. Related Work

The concept of MEC has been put forward many years ago. In 2013, the world's first MEC platform was established by IBM and Nokia Siemens Network [16]. In 2014, the European Telecommunications Standards Institute (ETSI) proclaimed industry specifications for MEC, which has been supported by IBM, Huawei, Intel, etc., [17]. As a main evolutional technology in the 5G communication system, the MEC technology has been greatly developed and gradually applied to many fields in recent years [18], [19].

Most of the MEC research works focus on how to fully utilize the powerful computing and storage capacity of the MEC server. The latency and energy consumption are improved by offloading large computational tasks to MEC server and precaching popular videos in MEC server [20]. Ketykó *et al.* [21] provided a general model of the system considering the computational latency of MEC applications and presented the solution method related with deployment, resource sharing, load balance, and fairness among multiple users. The joint optimization problem of computation offloading, resource allocation, and task proactive caching is considered to minimize the execution latency [22]. Zhang *et al.* [23] considered the energy-latency tradeoff problem in single and multicell MEC networks. The residual energy of smart devices' battery was introduced into the definition of the weighting factor of energy consumption and latency. Dinh *et al.* [24] proposed a framework where the mobile device minimized both its energy consumption and its tasks' execution latency by jointly optimizing the task allocation decision and its CPU frequency. Wu *et al.* [12] designed a joint optimization problem of mobile user's offloaded workloads and transmission time for nonorthogonal multiple access.

With the development of AR/VR, there are more and more researches on the AR task offloading. Liu *et al.* [25] designed an edge network orchestrator to enable fast and accurate object analytics at the network edge for mobile AR (MAR), and proposed a joint optimization algorithm of server assignment and frame resolution selection to tradeoff the service latency and analytics accuracy. Sun *et al.* [26] designed an implementation framework for mobile VR delivery by utilizing the caching and computing capabilities of mobile VR devices to alleviate the traffic burden over wireless networks. Akter *et al.* [27] presented MEC-based mixed reality (MR) application for assisting blind and visually impaired people. The computational task was offloaded to the nearest MEC server to prolong the battery life of the MR devices.

Generally speaking, there are two task offloading models: 1) binary offloading and 2) partial offloading. The binary offloading scheme decides whether a task should be offloaded to the edge server or executed locally as a whole. The above-related work is binary offloading. These algorithms are simple and resources cannot be fully utilized. The partial offloading scheme is to split a task into several subtasks and partially offload them to the edge server. Chai *et al.* [28] partitioned a

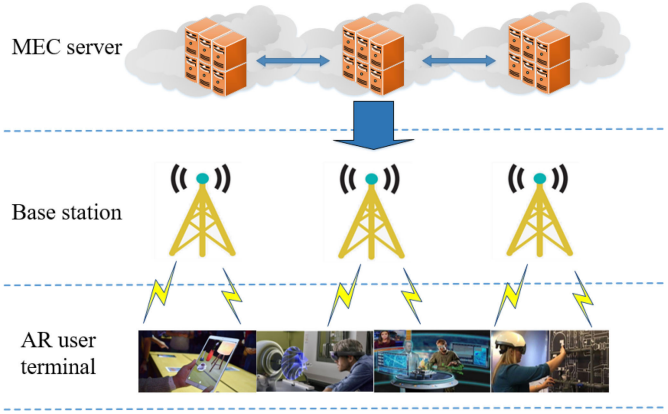

Fig. 1. Multi-MEC system model.

task into small portions with arbitrary data size, and developed a joint computing scheme of local computing, MEC offloading, and device-to-device offloading. Wang *et al.* [29] investigated the partial computation offloading by jointly optimizing the computational speed, transmit power, and offloading ratio. Yang *et al.* [30] considered each VR task that consists of several chunks, and proposed a task scheduling strategy to decide which computation model should be operated to minimize the communication-resource consumption under the delay constraint. Liu and Zhang [31] partitioned a task into subtasks and offloaded them to multiple nearby edge nodes. The tradeoff between the latency and reliability in task offloading to MEC was studied. Hu *et al.* [32] partitioned the unmanned aerial vehicle (UAV) computation task into two parts: 1) local computing and 2) MEC computing. The objective was to solve the joint optimization problem of UAV position, time slot allocation and computation task partition in order to minimize the system energy consumption of all users. The partial offloading models in the above literature are that a task can be arbitrarily divided into several subtasks, which is too ideal in reality. Most tasks cannot be arbitrarily divided according to their internal component functions in fact.

In practical application, there are two cases of partial offloading: 1) the task data are divided into several parts, which are offloaded to different MEC servers and processed at the same time. The above-mentioned literatures adopt this case and 2) according to the different functional characteristics in the process of task execution, the task execution process is divided into several subtasks, and then executed in time sequence. Hao *et al.* [9] split the AR application into five subtasks according to the functional component, and proposed a joint optimization problem of task caching and offloading to minimize the energy consumption of all users, but the dependencies between tasks were not considered. This article adopts the latter case. In this article, an AR application is divided into five subtasks according to its internal component functions, and the dependencies among subtasks are considered.

III. System Model

In this article, we consider a scenario where U AR clients can simultaneously perform AR applications by offloading

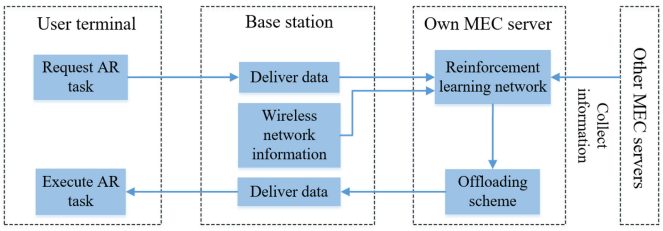

Fig. 2. Workflow diagram of the system.

some tasks to the MEC server in the single-MEC system and the multi-MEC system, respectively. A system model is proposed to minimize the energy consumption of each user while satisfying the delay requirement of AR application. We believe that as long as the processing time of the AR task is within the delay threshold, a satisfactory AR task experience can be obtained. It is not necessary to blindly reduce the processing delay of AR tasks. As shown in Fig. 1, the system is mainly composed of the MEC server, base stations, and user terminals. We denote the set of MEC server, the set of base stations, and the set of users by $\psi \in \{1, 2, \ldots, \Psi\}$, $n \in \{1, 2, \ldots, N\}$, and $u \in \{1, 2, \ldots, U\}$, respectively. $\Psi = 1$ denotes the single-MEC server system, $\Psi > 1$ denotes the multi-MEC server system. Let F_{ψ}^{MEC} denotes the computing resources of MEC server ψ. $\psi = 1$ denotes the own MEC server, and other values denote adjacent MEC servers. The framework relationship is that one MEC server covers multibase station and one base station covers multiuser terminals. In the multi-MEC server system, MEC servers can cooperate to execute tasks. The MEC server has a powerful computing capacity, which can quickly process offloaded tasks. The base station undertakes the forwarding function of tasks. We assume that each user terminal only performs one AR application in each decision process. A decision is made when the video frame is captured by the camera. The specific workflow of the system is depicted in Fig. 2. First, user terminals request the AR applications and send the relevant information to the MEC server through the base stations. Second, these users' own MEC server collects network status information and the adjacent MEC servers information to make the offloading and resource allocation schemes. Third, the own MEC server sends the offloading paths and resource allocation schemes to user terminals. Finally, user terminals execute the AR tasks according to the made offloading and resource allocation schemes. The main work of this article is how to obtain the offloading paths and resource allocation schemes before tasks are executed. Therefore, a joint optimization algorithm of resource allocation and task offloading based on deep reinforcement learning is proposed, and the details are shown in Section IV. The section of the system model consists of three parts: 1) AR task model; 2) communication model; and 3) computation model, which are described as follows.

A. AR Application Model

In order to utilize resources rationally, one AR application is divided into five subtasks: 1) the video source, which obtains

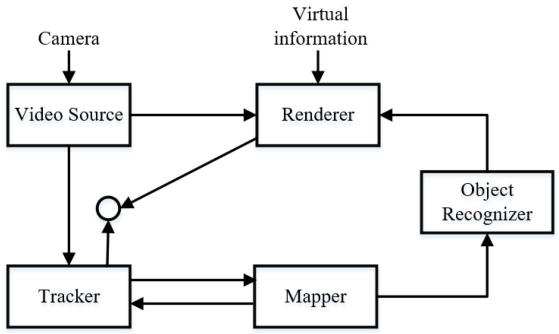

Fig. 3. AR application model.

raw video frames from the camera; 2) tracker, which tracks the position of the user concerning the environment; 3) mapper, which builds a model of the environment; 4) object recognizer, which identifies known objects in the environment; and 5) renderer, which prepares the processed frames for display, and plays the critical role in AR application [33]. The video source and renderer components must be executed locally result from the collected and displayed data can only be done at user terminals. Fig. 3 illustrates five subtasks of the AR task model. The scenario that the time interval between frames is less than the delay threshold is considered. In other words, the current frame has not been fully processed and the next frame has been captured. A subtask is executed after the end of the previous subtask. The execution results of the previous subtask need to be passed to the next subtask. The resources assigned to each subtask are dedicated until the subtask is completed. We denote an AR application by $m \in \{1, 2, \ldots, M\}$. Different AR applications have different delay thresholds. $k \in \{1, 2, 3, 4, 5\}$ indicates the kth subtask of AR application. $m_u \in \{1, 2, \ldots, M\}$ denotes the requested AR application by user u. To describe the parametric context of each subtask, we define a tuple representation as $\phi_m^k = (\omega_m^k, \varphi_m^k)$. Specifically, ω_m^k and φ_m^k are the input data size and computing workload of the kth subtask of AR application, respectively. The relationship between ω_m^k and φ_m^k is expressed as $\varphi_m^k = \eta_m^k \cdot \omega_m^k$, where η_m^k denotes the computing workload per bit [34]–[36]. As long as the data size of the first subtask is known, the data size of all subsequent subtasks can be known according to the proportion $\varpi \in [0, 1]$ and $\omega_m^k = \varpi \cdot \omega_m^{k-1}$. Delay plays an important role in the experience of AR application, so the time delay limit T_u is set. If the time delay of the AR application exceeds T_u, a good experience of user u cannot be achieved.

The notations mainly used throughout this article are summarized in Table I.

TABLE I
PARAMETER DESCRIPTIONS

n	Index of base station
ψ	Index of MEC server
u	Index of user terminal
k	The k-th subtask of AR application
m	Index of AR application
N	Number of base stations
U	Number of users
M	Numbe of AR applications
B	System bandwidth
ϖ	The proportion between the input and output data size of the subtask
m_u	The requested AR application by user u
F_ψ^{MEC}	The computing resources of MEC server ψ
Ψ	Number of MEC servers
k	The k-th subtask of AR application
ω_m^k	The output data size of k-th subtask of AR application m
φ_m^k	The computing workload of k-th subtask of AR application m
T_u	The delay limitation of AR application requested by user u
η_m^k	The computation-to-volume ratio of k-th subtask of AR application m
p_u^{UT}	The transmission powers of user terminal u
p_n^{BS}	The transmission powers of base station n

B. Communication Model

In this article, we consider the system with OFDMA as the multiple access technology, in which the system bandwidth B is divided into U equal subbands. In order to ensure the orthogonality of transmission link among users associated with the same BS, each user is assigned to a subband. Let p_u^{UT} and p_n^{BS} denote the transmission powers of user terminal u and base station n, respectively. h_u^{up} and h_u^{down} denote the uplink and downlink channel gains between the base station and user u, respectively. Then, the uplink and downlink transmission rates can be calculated by

$$r_{u,\vartheta}^{\mathrm{up}} = B_{u,\vartheta}^{\mathrm{up}} \log_2\left(1 + \frac{p_u^{\mathrm{UT}} h_u^{\mathrm{up}}}{\delta^2}\right) \tag{1}$$

$$r_{u,\vartheta}^{\mathrm{down}} = B_{u,\vartheta}^{\mathrm{down}} \log_2\left(1 + \frac{p_u^{\mathrm{BS}} h_u^{\mathrm{down}}}{\delta^2}\right) \tag{2}$$

where δ^2 denotes the noise power; $\vartheta \in \{1, 2, 3, 4\}$ denotes the transmission link between subtasks, i.e., $\vartheta = 1$ denotes the uplink between subtask 1 and 2, and there is no downlink because subtask 1 can only be performed locally; $\vartheta = 2$ denotes the uplink and downlink between subtask 2 and 3; and $B_{u,\vartheta}^{\mathrm{up}}$ and $B_{u,\vartheta}^{\mathrm{down}}$ denote the uplink and downlink channel bandwidths allocated to ϑth link of user u, which are constrained by

$$\sum_{\vartheta} B_{u,\vartheta}^{\mathrm{up}} \le B_u^{\mathrm{up}} = \frac{B\alpha}{U} \tag{3}$$

$$\sum_{\vartheta} B_{u,\vartheta}^{\mathrm{down}} \le B_u^{\mathrm{down}} = \frac{B\hat{\alpha}}{U} \tag{4}$$

where α and $\hat{\alpha} \in [0, 1]$ denote the utilization rate of uplink and downlink system bandwidth, respectively. Therefore, the time delay $t_{u,\vartheta}^{\mathrm{up}}$ and the terminal energy consumption $e_{u,\vartheta}^{\mathrm{up}}$ of uplink transmission can be calculated by

$$t_{u,\vartheta}^{\mathrm{up}} = \frac{\omega_{m_u}^{\vartheta}}{r_{u,\vartheta}^{\mathrm{up}}} \tag{5}$$

$$e_{u,\vartheta}^{\mathrm{up}} = \omega_{m_u}^{\vartheta} \cdot \rho \tag{6}$$

where ρ denotes the energy consumption per sent bit at the user terminal. The time delay $t_{u,\vartheta}^{\mathrm{down}}$ of downlink transmission is calculated by

$$t_{u,\vartheta}^{\mathrm{down}} = \frac{\omega_{m_u}^{\vartheta}}{r_{u,\vartheta}^{\mathrm{down}}}. \tag{7}$$

The energy consumption of the user terminal receiving data is not considered in this article.

C. Computation Model

In this article, we focus on the collaborative task execution between the user terminal and MEC servers. Specifically, each subtask except the video source and renderer can be executed locally or offloaded to MEC server, based on the allocated computing resource and communication resource of the subtask. Next, three situations are described in detail, respectively.

1) Processing at Own MEC: Own MEC indicates that the user is directly served by the MEC. Let $f_{1,u}^k$ denote that the computing resources of kth subtask allocated by F_1^{MEC} to user u. If kth subtask of user u is executed on own MEC server, the time delay $t_{1,u,k}^{\text{MEC}}$ can be calculated by

$$t_{1,u,k}^{\text{MEC}} = \frac{\varphi_{m_u}^k}{f_{1,u}^k}. \tag{8}$$

2) Processing at Adjacent MEC: If own MEC does not have enough computing resources for processing the subtask and the computing resources of adjacent MEC are rich, the computing resources will be allocated to the subtask by the adjacent MEC. Let $f_{\psi_{-1},u}^k$ denote that the computing resources of kth subtask allocated by $F_{\psi_{-1}}^{\text{MEC}}$ to user u. ψ_{-1} denotes adjacent MEC servers except for own MEC server. If kth subtask of user u is executed on adjacent MEC server ψ_{-1}, the time delay $t_{\psi_{-1},u,k}^{\text{MEC}}$ can be calculated by

$$t_{\psi_{-1},u,k}^{\text{MEC}} = \frac{\varphi_{m_u}^k}{f_{\psi_{-1},u}^k} + \frac{d_{\psi_{-1}}}{\xi} \tag{9}$$

where $d_{\psi_{-1}}$ is the distance from the adjacent MEC to own MEC, and ξ is the transmission speed of the wired link. Since the communication way between MEC servers is through the wired link, the limitation of transmission bandwidth between MEC servers is not considered. If two adjacent subtasks are offloaded to different MEC servers, the output of the previous subtask needs to be passed to the MEC server where the next subtask is located through a wired connection.

3) Processing at User Terminal: The total computing resource of user u is defined by F_u^{UT}. Let h_u^k denote that the computing resource of kth subtask allocated by the terminal device of user u. If kth subtask of user u is executed on the user terminal, the time delay $t_{u,k}^{\text{UT}}$ and the terminal energy consumption $e_{u,k}^{\text{UT}}$ can be calculated by

$$t_{u,k}^{\text{UT}} = \frac{\varphi_{m_u}^k}{h_u^k} \tag{10}$$

$$e_{u,k}^{\text{UT}} = \kappa \cdot (h_u^k)^2 \cdot \varphi_{m_u}^k \tag{11}$$

where κ is the energy coefficient, which depends on the chip architecture. In this article, according to the work in [9], we set $\kappa = 10^{-25}$.

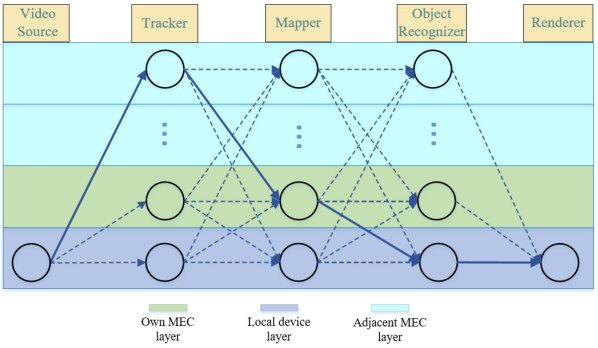

Fig. 4. AR application graph.

IV. Joint Optimization of Task Offloading and Resource Allocation

In this section, the joint optimization problem of task offloading and resource allocation is formulated, and it is considered as MDP. A deep reinforcement learning algorithm based on MADDPG framework is proposed to solve this problem. The details of problem formulation and solution are described as follows.

A. Problem Formulation

According to the above analysis, the executing processes of AR application are modeled as a weighted directed graph $\mathcal{G}_u = (\mathcal{V}_u, \mathcal{L}_u) = (\{\mathcal{V}_{\psi,u}^{\text{MEC}}, \mathcal{V}_u^{\text{UT}}\}, \{\mathcal{L}_u^{\text{up}}, \mathcal{L}_u^{\text{down}}, \mathcal{L}_u^{\text{hold}}\})$, as shown in Fig. 4. $\mathcal{V}_u$ and $\mathcal{L}_u$ denote the set of vertex v and arc l in $\mathcal{G}_u$, respectively. $\mathcal{V}_{\psi,u}^{\text{MEC}}$ and $\mathcal{V}_u^{\text{UT}}$ denote the set of vertex of MEC ψ layer and local user terminal layer, respectively. $\mathcal{L}_u^{\text{up}}$, $\mathcal{L}_u^{\text{down}}$, and $\mathcal{L}_u^{\text{hold}}$ denote the set of uplink, downlink, and no-link. No-link means that two adjacent subtasks are executed in the same place. g_u is defined as a set of vertex of a path that can complete the AR application, and consists of $v_{g_u}^{\text{UT}}$ and $v_{\psi,g_u}^{\text{MEC}}$. $v_{g_u}^{\text{UT}}$ and $v_{\psi,g_u}^{\text{MEC}}$ denote a set of vertex of local user terminal layer and MEC ψ layer in the path g_u, respectively.

Therefore, the total time delay and the total energy consumption of user u can be calculated by

$$t_{g_u} = \sum_{s,c \in g_u} \left(t_u^{s,c}\right) + \sum_{v \in g_u} \left(t_{u,v}\right) \tag{12}$$

$$e_{g_u} = \sum_{\substack{s \in v_{g_u}^{\text{UT}} \\ c \in v_{g_u}^{\text{MEC}}, v_{\psi_{-1},g_u}^{\text{MEC}}}} e_u^{s,c} + \sum_{v \in g_u} e_{u,v} \tag{13}$$

where s and c denote the start and end vertex of an arc. $t_u^{s,c}$ and $e_u^{s,c}$ denote the transmission time and energy consumption from vertex s to c, respectively. $t_{u,v}$ and $e_{u,v}$ denote the execution time and energy consumption at vertex v, respectively.

Next, we will provide the formulated problem of joint optimization of resource allocation and task offloading. The mathematical model with the objective of minimizing the energy consumption of each user terminal, subject to system resource and delay threshold, is as follows:

$$\begin{aligned}
\min_{B_{u,l}^{\text{up}}, B_{u,l}^{\text{down}}, f_{\psi,u}^{v}, h_u^v, g_u} \quad & e_{g_u} \quad \forall u \in \{1, 2, \ldots, U\} \\
\text{s.t.} \quad & (c1)\ t_{g_u} \leq T_u \\
& (c2)\ \sum_{l \in \mathcal{L}_u^{\text{up}}} B_{u,l}^{\text{up}} \leq B_u^{\text{up}} \\
& (c3)\ \sum_{l \in \mathcal{L}_u^{\text{down}}} B_{u,l}^{\text{down}} \leq B_u^{\text{down}} \\
& (c4)\ \sum_{u} \sum_{v \in \mathcal{V}_{\psi,u}^{\text{MEC}}} f_{\psi,u}^{v} \leq F_{\psi}^{\text{MEC}} \\
& \qquad \psi \in \{1, 2, \ldots, \Psi\} \\
& (c5)\ \sum_{v \in \mathcal{V}_u^{\text{UT}}} h_u^v \leq F_u^{\text{UT}}.
\end{aligned} \tag{14}$$

For the constraints, constraint (C1) indicates that the execution delay of the AR application cannot exceed the delay limit to ensure the service experience quality. For a single MEC system, the delay does not include the execution time consumed in the adjacent MEC servers. Constraint (C2) means that the total uplink transmission resource allocated to the five subtasks cannot exceed the uplink transmission resource of the user. Constraint (C3) means that the total downlink transmission resource allocated to the five subtasks cannot exceed the downlink transmission resource of the user. Constraint (C4) means that the sum of computing resources of five subtasks allocated to all users cannot exceed the computing resource of the MEC server. When $\Psi = 1$, the problem (14) is the single-MEC mathematical model. When $\Psi > 1$, problem (14) is the multi-MEC mathematical model. Constraint (C5) means that the sum of five subtasks cannot exceed the computing resource of the user.

B. Deep Reinforcement Learning-Based Solution

Problem (14) is a multiuser cooperation and competition problem, which is an NP-hard problem. In this section, we propose a joint optimization algorithm of resource allocation and task offloading based on deep reinforcement learning.

Reinforcement learning is one of the paradigms and methodologies of machine learning. It is used to describe and solve the problem that agents learn strategies to achieve specific goals in the process of interaction with the environment. It is different from supervised learning, which is mainly manifested in reward signals. Reward signals are used to evaluate the quality of actions, rather than telling the reinforcement learning system how to produce correct actions. In recent years, reinforcement learning is widely used in information theory, game theory, automatic control, and other fields. Since the scenario of this article is a game theory problem of multiuser competition and cooperation, we consider using reinforcement learning to solve it. The MADDPG algorithm [37]–[39] is a paradigm of the reinforcement learning method. It is an improvement of the deep deterministic policy gradient (DDPG) algorithm [40] to adapt to a multiagent environment, and its core idea is centralized training and decentralized execution. The common model of reinforcement learning is the standard MDP. Therefore, we consider the joint optimization problem as MDP, which is modeled by the MADDPG framework.

1) State Space: State is the description of the environment, which will change after an action is generated by the agent. Let $S_t = (s_1, s_2, \ldots, s_U)$ denote the state of MDP at time t, which contains all parameters of AR application, resources allocation, and network conditions. Specifically, according to the multi-MEC server system, if the number of users and MEC servers is assumed to be 15 and 3, respectively, $s_u = (s_{u,1}, s_{u,2}, \ldots, s_{u,68})$ denotes the state of user u, which is composed of 68 dimensional data (e.g., s_1 and a_1 are shown in Fig. 5). $s_{u,2}, \ldots, s_{u,15}$ denotes the allocation result of communication and computing resource of user u at the previous time. In the single-MEC server system, data related to adjacent MEC servers $s_{u,16}, \ldots, s_{u,21}$, $s_{u,23}$, $s_{u,24}$, $a_{u,15}, \ldots, a_{u,20}$ are not required. $a_{u,1}, \ldots, a_{u,20}$ is the change amount of resource allocation result $s_{u,2}, \ldots, s_{u,21}$ for multi-MEC servers. In this article, we assume that the input and output of each subtask have a certain proportion. Therefore, only the data size of raw video frames from the camera is needed, and the input data size of other subtasks is not needed. To solve the problem of network overflow, the state values need to be normalized to the same order of magnitude. The value range of all data in the state is [0, 1]. Data 2-15 of the state indicate the proportion of allocated resources to the total resources.

2) Action Space: Action is the description of agent behavior, which is the result of agent decision. Let $A_t = (a_1, a_2, \ldots, a_U)$ denote the action of MDP at time t, which includes the change of computing and transmission resources, as shown in Fig. 5. In this article, the action is to increase or decrease the current state of resources allocation. The value range of all data in the action is $[-1, 1]$.

3) Reward: After the agent takes an action, reward is the feedback of environment to agent. In this specific scenario, the meaning of the reward is the amount of energy consumed by the user terminal after executing the resources allocation and offloading scheme according to the action. Let $R_t = (r_1, r_2, \ldots, r_U)$ denote the reward of MDP at time t. The purpose of this article is to minimize the energy consumption of each user terminal, subject to the system resource and delay threshold. Therefore, the reward is set according to the constraints and objective, including the reward of task completion, delay constraint, and energy consumption, which are pursued in sequence.

1) *Task Completion:* Ensure that the allocated resources enable AR application to be completely executed. As shown in the solid line of Fig. 4, there is a complete path from the video source to the renderer.
2) *Delay Constraint:* Ensure that the execution delay of AR application does not exceed the delay threshold.
3) *Energy Consumption:* The lower the energy consumption is, the higher the reward is.

In order to explain our reward more intuitively, the reward of resources constraints is designed as

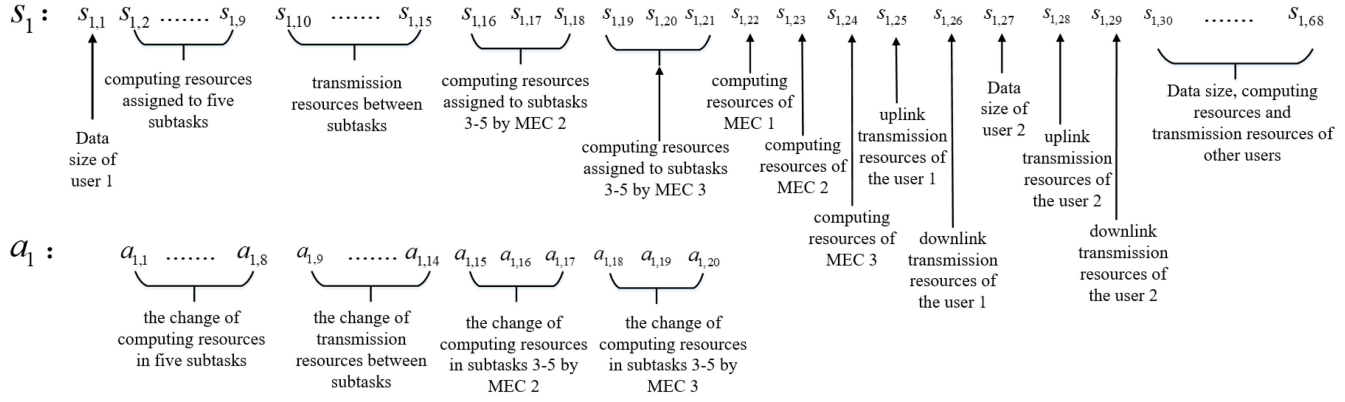

Fig. 5. State and action of user 1 (the number of users is 15).

$$
\begin{aligned}
r_u = {} & \chi_1 \cdot \sum_{q=2}^{15} \big((s_{u,q}-1)\cdot\varepsilon(s_{u,q}-1) + s_{u,q}\cdot\varepsilon(-s_{u,q})\big) \\
& + \chi_2 \cdot \left(\sum_{l\in\mathcal{L}_u^{\mathrm{up}}} B_{u,l}^{\mathrm{up}} - B_u^{\mathrm{up}}\right)\cdot\varepsilon\left(\sum_{l\in\mathcal{L}_u^{\mathrm{up}}} B_{u,l}^{\mathrm{up}} - B_u^{\mathrm{up}}\right) \\
& + \chi_3 \cdot \left(\sum_{l\in\mathcal{L}_u^{\mathrm{down}}} B_{u,l}^{\mathrm{down}} - B_u^{\mathrm{down}}\right) \\
& \times \varepsilon\left(\sum_{l\in\mathcal{L}_u^{\mathrm{down}}} B_{u,l}^{\mathrm{down}} - B_u^{\mathrm{down}}\right) \\
& + \chi_4 \cdot \sum_{\psi}\left(\left(\sum_{u}\sum_{v\in\mathcal{V}_{\psi,u}^{\mathrm{MEC}}} f_u^v - F_\psi^{\mathrm{MEC}}\right)\right. \\
& \qquad \left.\times\,\varepsilon\left(\sum_{u}\sum_{v\in\mathcal{V}_{\psi,u}^{\mathrm{MEC}}} f_u^v - F_\psi^{\mathrm{MEC}}\right)\right) \\
& + \chi_5 \cdot \left(\sum_{v\in\mathcal{V}_u^{\mathrm{UT}}} h_u^v - F_u^{\mathrm{UT}}\right)\cdot\varepsilon\left(\sum_{v\in\mathcal{V}_u^{\mathrm{UT}}} h_u^v - F_u^{\mathrm{UT}}\right) + b_1
\end{aligned}
\tag{15}
$$

where $\varepsilon()$ is a step function, the reward of experience quality is designed as

$$
r_u = \max_{g_u}\left\{\chi_6\cdot\left(t_{g_u} - T_u\right)\cdot\varepsilon\left(t_{g_u} - T_u\right) + b_2\right\} \tag{16}
$$

the reward of energy consumption model is designed as

$$
r_u = \max_{g_u}\left\{\chi_7\cdot\exp\left(-e_{g_u}\right) + b_3\right\} \tag{17}
$$

where χ_1, χ_2, χ_3, χ_4, χ_5, χ_6, χ_7, b_1, b_2, and b_3 are constants. The purpose is to make rewards develop in a good direction. Specifically, the reward setting algorithm is illustrated in Algorithm 1.

Algorithm 1 Reward Calculation Algorithm

Input: new state $S_{t+1} \leftarrow$ *environment*(S_t, A_t)
Output: R_t
Begin
Initialize the reward R_t=[]
for user $u = 1, 2, \ldots, U$
  if $0 \le s_{u,2}, s_{u,3}, \ldots, s_{u,15} \le 1$ and C2, C3, C4, C5 of (14)
    if task is completely executed
      for each path g_u
        if C1 of (14)
          Calculate the reward of energy consumption according to equation (17)
        else
          Calculate the reward of experience quality according to equation (16)
        end
      end
      The value of r_u is the highest reward in all paths
    else
      $r_u = b_4$
    end
  else
    Calculate the reward of resources constraints according to equation (15)
  end
end

4) MADDPG: In this section, the network framework and the implementation process of the algorithm are introduced. The reasons that the problem (14) is modeled by MADDPG are as follows: 1) according to the definition of action space, the action is deterministic and continuous and 2) the joint optimization is the mixed cooperative-competitive environment. In general, MADDPG is the combination of DQN and actor–critic (AC). The MADDPG learning framework adopts the AC method, which consists of two part: 1) actor network and 2) critic network. Each actor and critic network consists of target network and evaluation network, as shown in Fig. 6. The actor network is used to generate user's action and the critic network is used to criticize the action. Let μ_u and μ_u' denote the evaluation network with parameter θ_u and the target network with parameter θ_u' of actor network. Q_u and Q_u' denote the evaluation network with parameter θ_u^Q and the target network with parameter $\theta_u'^Q$ of the critic network.

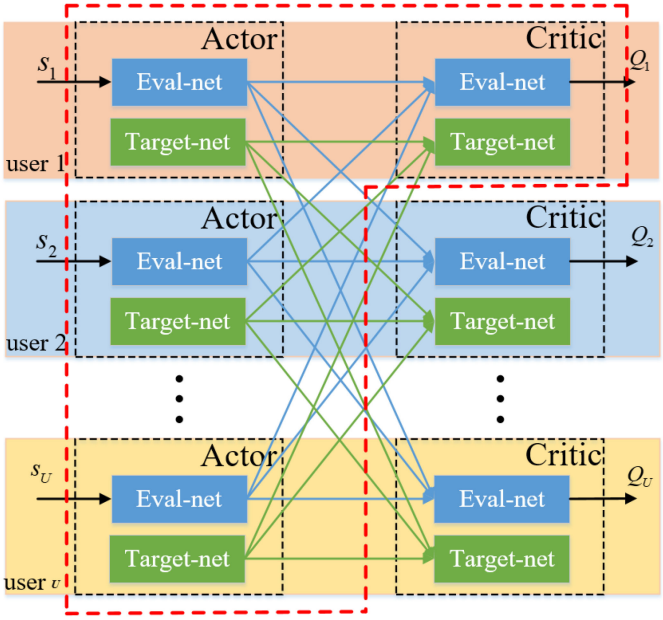

Fig. 6. Network structure of MADDPG.

Algorithm 2 Training Process

```
for episode t = 1, 2, ..., M
    Initial state S
    for each user u = 1, 2, ..., U
        Select action a_u = μ_u(s_u) + N, N is an
        exploration noise
        Execute actions A_t = (a_1, ..., a_U) and observe reward
        R_t and new state S_{t+1}
        Store (S_t, A_t, S_{t+1}, R_t) in replay buffer Φ
    end
    for each user u = 1, 2, ..., U
        Sample a random mini-batch of X samples
        (S^X, A^X, S'^X, R^X) from Φ
        Update critic by minimizing the loss Equation (18)
        Update actor using the sampled policy gradient:
        Equation (20)
    end
    Update target network parameters for each user u:
        θ'_u = τθ_u + (1 − τ)θ'_u
end
```

In this article, the initial resource allocation of each user is imputed into the actor network to generate a resource allocation adjustment. Then, the initial state and the adjustment of all users are imputed into the critic network to obtain a Q value as the feedback to the adjustment. The reason that the critic network needs to input the states and actions of all users is because there is a relationship of competition and cooperation between users, and the action of each user needs to be considered. When the network training is finished, the result of adjusting the initial resource allocation is the current resource allocation scheme. At training time t, the data (S_t, A_t, S_{t+1}, R_t) are stored in the replay buffer Φ according to the format of (S, A, S', R). The system starts training until there is enough data in Φ. During the training process, a minibatch data (S^X, A^X, S'^X, R^X) will be extracted from Φ for training. The critic network is trained according to the estimated Q value and the actual Q value as follows:

$$\text{Loss}_u = \frac{1}{X}\sum_{j=1}^{X}\left(y_u^j - Q_u(S^j, a_1^j, a_2^j, \ldots, a_U^j)\right)^2 \tag{18}$$

$$y_u^j = r_u^j + \gamma Q_u'\left(S'^j, a_1'^j, \ldots, a_U'^j\right)\Big|_{a_k'^j=\mu_u'\left(s_k'^j\right)} \tag{19}$$

where X denotes the size of minibatch data, $S^j = \{s_1^j, s_2^j, \ldots, s_U^j\}$ denotes the jth state of minibatch data, $S'^j = \{s_1'^j, s_2'^j, \ldots, s_U'^j\}$ denotes the jth new state of minibatch data, and γ denotes the discount factor. The actor network is updated according to the feedback of the critic network as follows:

$$\nabla_{\theta_u} J \approx \frac{1}{X}\sum_{j=1}^{X}\nabla_{\theta_u}\mu_u\left(s_u^j\right)\nabla_{a_u}Q_u \times \left(S^j, a_1^j, \ldots, a_u, \ldots, a_U^j\right)\Big|_{a_u=\mu_u\left(s_u^j\right)}. \tag{20}$$

In the common reinforcement learning training process, there is a greedy strategy to balance exploration and exploitation. Similarly, in MADDPG, the random numbers of a normal distribution are introduced as the exploration noise. MADDPG has the characteristics of centralized training and decentralized execution. When testing, we only need the actor network to complete the resource allocation without the critic network. The training process is summarized in Algorithm 2. The term "episode" means the number of epochs used in the training process.

V. Performance Evaluation

A. Parameter Setting

In this section, we evaluate the performance of our proposed algorithm for AR application in the MADDPG system framework. The experimental platform adopts DELL PowerEdge (DELL-R940XA, 4*GOLD-5117, RTX2080Ti). In MADDPG, neural networks with two hidden layers are trained, and its activation is a RELU function. Tanh function is used in the output layer of the actor network to constraint the output action value. Specifically, the simulation parameters of the system are presented in Table II. The compared algorithms are as follows.

1) *Single-MEC:* Joint optimization of resource allocation and task offloading based on deep reinforcement learning for single-MEC server systems in this article.
2) *Multi-MEC:* Joint optimization of resource allocation and task offloading based on deep reinforcement learning for multi-MEC server systems in this article.
3) *MASL:* The multiagent stochastic learning algorithm is proposed in [41]. The multiagent computation offloading problem is modeled as a stochastic game. The flow of the algorithm is as follows.
    a) Each user sets its strategy selection probability vector as a uniform distribution. Since the VR application is divided into five subtasks, there

TABLE II
PARAMETER CONFIGURATION

Parameter	value
Number of users	15
Number of base stations	3
Number of MEC servers	3
Uplink/Downlink system Bandwidth	75MHz
Transmission powers of user terminal	0.5W
Transmission powers of base station	1W
Noise power	-100dB
Size of video frame	[15, 35]Kb
Computing workload density	[10, 1000] CPU cycles/bit
Path loss model	$PL = 127 + 30log(dis)$
Energy consumption per sent Mbit	1.87J/Mb
Computing resources of local device	[1.8, 2.2]GHz
Computing resources of MEC server	[5, 30]GHz
Delay threshold of AR application	[25, 30]ms
episode	140000
Min batch	100
Buffer size	10000
Critic network learning rate	0.001
Actor network learning rate	0.0001
Optimizer	Adam

are eight offloading paths as the set of strategy selection.

b) Each user selects an offloading strategy according to the strategy selection probability vector.

c) Each user evaluates its respective energy consumption and the corresponding reward according to the selected offloading strategy.

d) Each user updates its strategy selection probability vector according to the reward until it does not adjust.

4) *GREEDY:* The greedy algorithm is designed according to [42] and [43]. The main procedures of this algorithm are as follows.

a) Subtasks 1 and 5 are initially executed on the user terminal, and subtasks 2, 3, and 4 are initially executed on the MEC server.

b) Resources are allocated proportionally according to the computing workload of subtasks.

c) If the delay requirement is not satisfied, subtasks 2, 3, and 4 are executed successively on the user terminal until the delay requirement is satisfied.

d) On the premise of ensuring the user's delay requirements, the computing resources allocated to subtasks by the user terminal should be reduced as much as possible.

5) *LOCAL:* In this algorithm, there is no MEC server. All subtasks of AR applications are executed on the user terminal, and computing resources are allocated proportionally according to the computing workload of subtasks.

6) *RANDOM:* The RANDOM algorithm is to randomly offload any tasks in the tracker, mapper, and object recognizer to the MEC server.

B. Performance of MADDPG Algorithm

Fig. 7 shows the convergence performance of the proposed algorithm with different learning rates. In this article, the

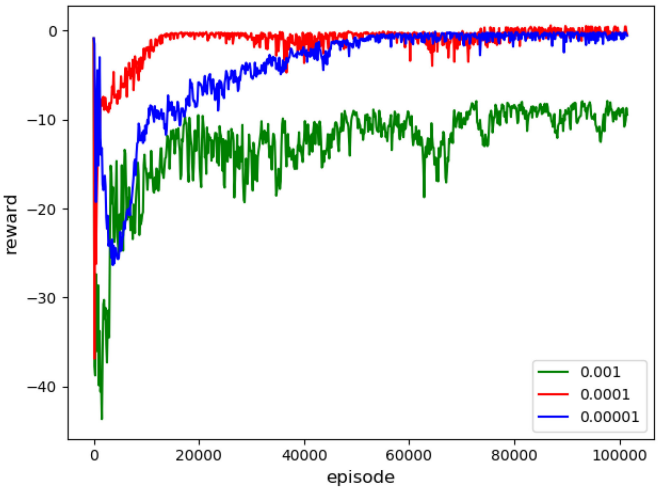

Fig. 7. Convergence property of different learning rates.

Adam optimizer is adopted to train the MADDPG network. The Adam optimizer is an adaptive learning rate optimization method, which uses the first-order moment estimation and the second-order moment estimation of the gradient to dynamically adjust the learning rate. From Fig. 7, we observe that when the learning rate is 0.0001, the network convergence performance is best. The network convergence speed is slower when the learning rate is 0.00001. The network training process cannot converge when the learning rate is 0.001. A higher learning rate will cause the network not to converge, and a lower learning rate will cause the network to converge slowly. Therefore, the learning rate of the training process is set to 0.0001.

Figs. 8 and 9 show the test performance of the proposed algorithm in terms of the reward and the energy consumption of all users. From Fig. 8, it is observed that the reward of each user in our algorithm (single-MEC and multi-MEC) is better. The reason is that the system communication environment is very unstable, and there exist competition and cooperation among agents. If the state of one agent changes, the state of other agents will be affected. The single-MEC algorithm has no cooperation with other MEC servers, so the reward is less than that of the multi-MEC algorithm. The average user energy consumption of the multi-MEC algorithm is 71% lower than that of the single-MEC algorithm. The improvement is more obvious when the user's own MEC has less available resources, as shown in Fig. 12. The average user energy consumption of the single-MEC algorithm is 35% lower than that of the MASL algorithm. The main reason is that the MASL algorithm only studies the offloading path with smaller space, but does not learn the resource allocation with larger space. The GREEDY algorithm cannot deal with the joint optimization problem of resource allocation and task offloading well in an unstable environment. The LOCAL algorithm means that all tasks are executed at the user terminal, and the reward must be the smallest. The RANDOM algorithm randomly selects some subtasks to offload, so the reward is higher than the LOCAL algorithm. MADDPG framework can obtain the actions of other agents, and make reasonable

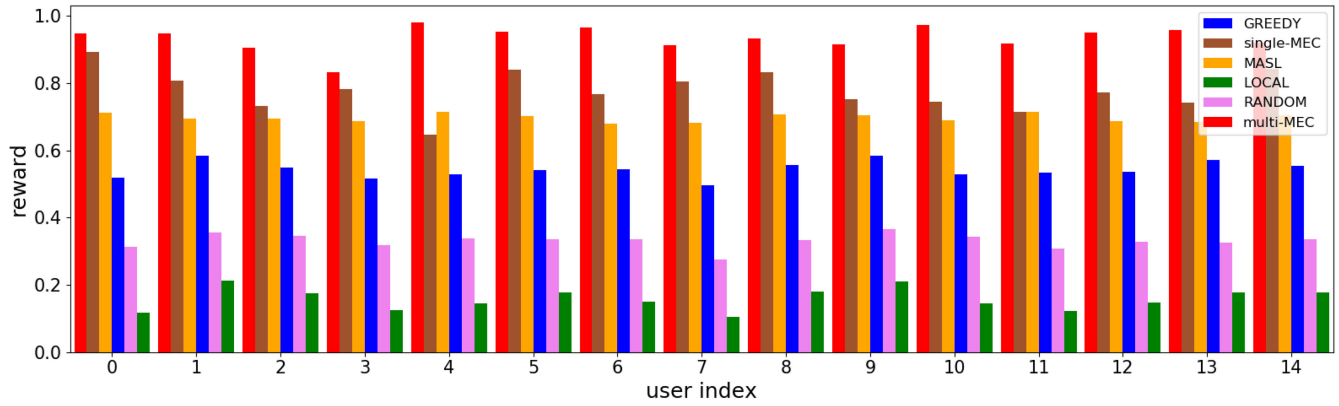

Fig. 8. Performance evaluation on reward.

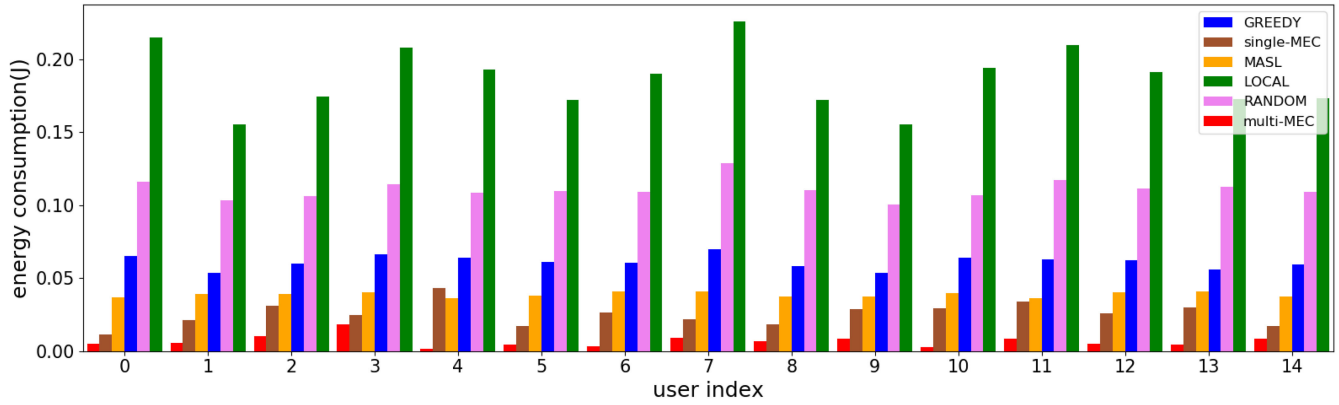

Fig. 9. Performance evaluation on energy consumption.

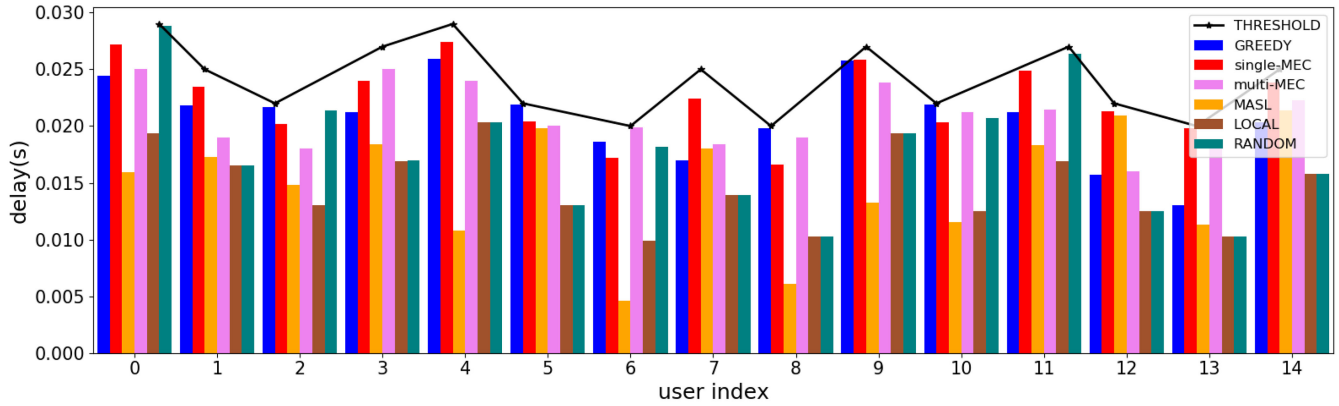

Fig. 10. Delay of each user in different algorithms.

decisions from a global perspective. The purpose of this article is to minimize the energy consumption of each user as much as possible, which is verified by Fig. 8. Fig. 9 shows the test performance of the energy consumption of each user. According to (17), it can be seen that the relationship between reward and energy consumption is an exponential function under the constraints of resource and delay. Therefore, Figs. 8 and 9 are one-to-one correspondence.

Fig. 10 shows the delay of six algorithms in the same environment configuration. The delay of the six algorithms is less than the delay threshold. Compared with the LOCAL algorithm, the delay of other algorithms is higher. The main reason is that the computing resources assigned by the MEC server to subtasks in other algorithms are less than that assigned by the local device in LOCAL algorithms in the case of a large number of users. In the conditions of satisfying the delay constraint, the energy consumption of our algorithms is reduced at the expense of delay.

C. Influence of Different Parameters

Since the aim of our algorithm is to minimize the energy consumption of each user terminal, the energy consumption of each user under different MEC computing resources is analyzed, as shown in Fig. 11. It can be seen that when own MEC

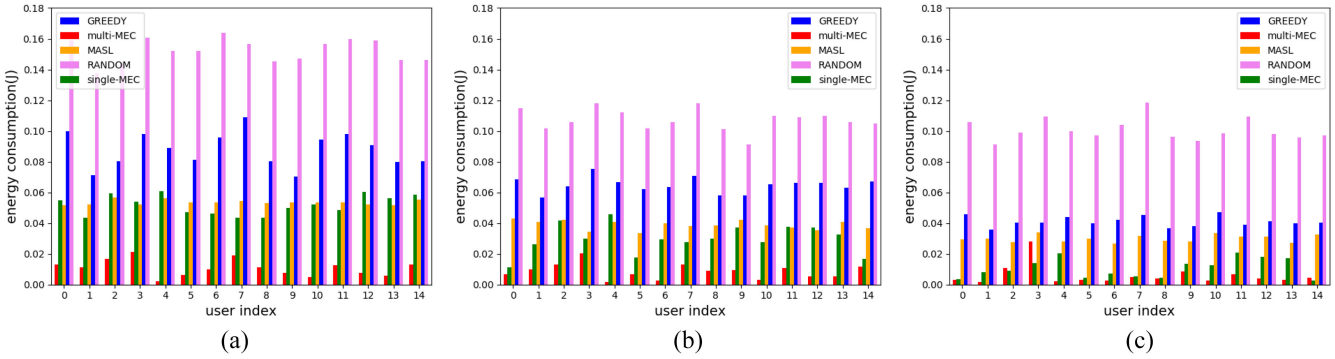

Fig. 11. There are the energy consumption of each user in different algorithms when the computing resources of own MEC are (a) 5 GHz, (b) 15 GHz, and (c) 25 GHz.

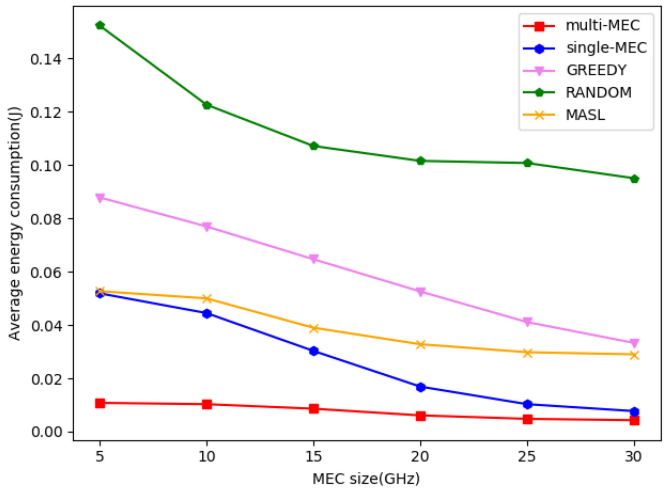

Fig. 12. Energy consumption of each user terminal with different MEC computing resources. The computing resources of own MEC are 5, 10, 15, 20, 25, and 30 GHz.

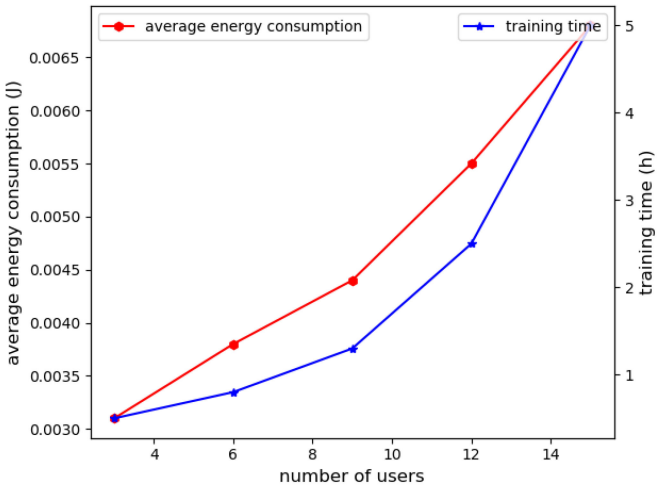

Fig. 13. Average energy consumption and the training time for different numbers of users. The numbers of users are 3, 6, 9, 12, and 15.

computing resources are 5, 15, and 25 GHz, our algorithms (single-MEC and multi-MEC) have better performance compared with the MASL, GREEDY algorithm and RANDOM algorithm. When the computing resources of own MEC server are 5, 15, and 25 GHz, the average user energy consumption of the multi-MEC algorithm is 79%, 71%, and 44% lower than that of the single-MEC algorithm. When the computing resources of own MEC are 5 and 15 GHz, the performance of the multi-MEC algorithm is better than that of the single-MEC algorithm due to the cooperation between MEC servers. But when the computing resource of own MEC is 25 GHz, several users' energy consumption in the multi-MEC algorithm is higher compared with the single-MEC algorithm, as shown in Fig. 11(c). The reason is as follows. When the computing resources of own MEC are sufficient, the offloading decision of all users is determined. The offloading scheme turns to the energy-efficient offloading mode: video source and renderer components are executed locally; and tracker, mapper, and object recognizer are executed on the own MEC server. Therefore, the offloading scheme of multi-MEC servers is close to that of the single-MEC server when its own MEC computing resources are sufficient. Since the environment of multi-MEC servers is more complex, the network training effect is poor compared with the single-MEC server. This theory is verified by Figs. 14(b) and 15(b). However, in practice, MEC computing resources are generally limited because of the equipment cost. The average user energy consumption of the single-MEC algorithm is 3%, 22%, and 64% lower than that of the MASL algorithm when the computing resources of own MEC server are 5, 15, and 25 GHz.

Here, we evaluate the effect of the size of MEC computing resources on energy consumption. Fig. 12 shows the average energy consumption when the computing resources of own MEC server are 5, 10, 15, 20, 25, and 30 GHz. With the increase of MEC computing resources, the average energy consumption decreases. This is mainly due to the increase of total MEC computing resources, resulting in the increase of MEC computing resources and the decrease of local computing resources allocated to each user, so the energy consumption of the user terminal is reduced.

From Fig. 12, we can find that the average energy consumption of the single-MEC algorithm is close to that of the multi-MEC algorithm when the MEC size is 30 GHz. When the MEC computing resources are sufficient, the offloading decision of all users is basically determined. Tracker, mapper, and object recognizer components are executed on their own MEC server. Therefore, the performance of the multi-MEC

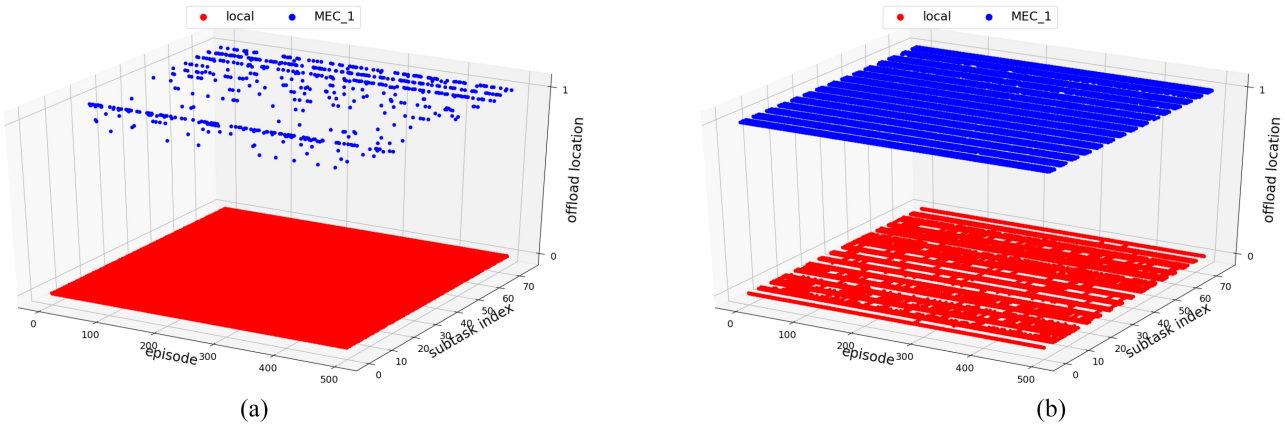

Fig. 14. Offloading location of AR task in single-MEC system. (a) MEC = 5 GHz. (b) MEC = 25 GHz.

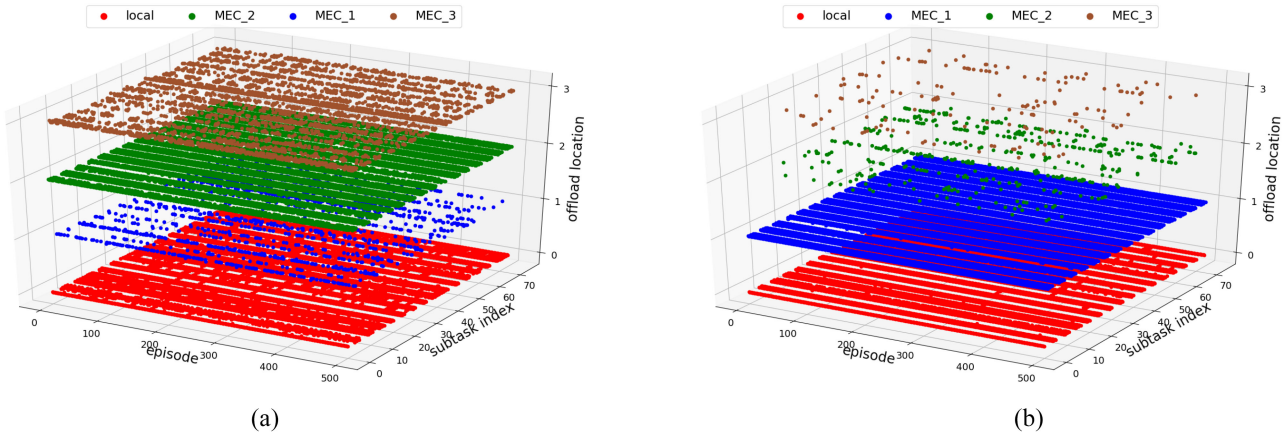

Fig. 15. Offloading location of AR task in multi-MEC system. (a) Own MEC = 5 GHz. (b) Own MEC = 25 GHz.

algorithm will be closer to that of the single-MEC algorithm when its own MEC computing resources are sufficient. The average energy consumption of the single-MEC algorithm is close to that of the MASL algorithm when the computing resources of own MEC server are 5 GHz. The reason is similar to the above. The case that all subtasks are executed on the local device becomes more frequently when the MEC computing resources are insufficient. For the simple offloading case, the MASL algorithm can achieve better performance by allocating resources according to the data proportion.

Fig. 13 shows the effect of the different numbers of users on energy consumption and training time. We can observe that with the increase of number of users, the average energy consumption and the training time increase. When the number of users increases, MEC computing resources allocated to each user are reduced. In order to meet the delay requirements, the local computing resources allocated to each user need to be increased, so the energy consumption of the user terminal is increased. For training time in Fig. 13, the increase in the number of users leads to an increase in the complexity of the environment.

D. Analysis of Offloading Location

Figs. 14 and 15 show the offloading location of subtasks in the single-MEC algorithm and the multi-MEC algorithm when own MEC computing resources are 5 and 25 GHz. The range of subtask index is [0, 74], meaning that 15 users have 75 subtasks in total. The value range of the offloading location is $\{0, 1, 2, 3\}$, values 0, 1, 2, 3 indicate that the subtask is offloaded to the local user terminal, MEC server 1, MEC server 2, and MEC server 3, respectively. Fig. 14(a) shows the offloading location of subtasks in the single-MEC system when the computing resources of own MEC server are 5 GHz. Since the computing resource of own MEC server are insufficient, most of the subtasks are executed on the local user terminal. When the computing resource of own MEC server is sufficient, subtasks (tracker, mapper, and object recognizer) are executed on their own MEC server, and other subtasks (video source and renderer) are executed on the local user terminal. As shown in Fig. 15(b), the red and blue dots present an interval display that validates the above description.

The irregular points in Fig. 14(b) are mainly caused by the fluctuation of the communication channel. When the MEC computing resource is 25 GHz, it is sufficient to meet the task offloading of all users. The ideal offloading state is that video source and renderer are executed on the local device, and tracker, mapper, and object recognizer are offloaded to the MEC server. But the wireless communication network will fluctuate in real time, which will affect the data transmission in the offloading process. Fig. 16(a) and (b) shows

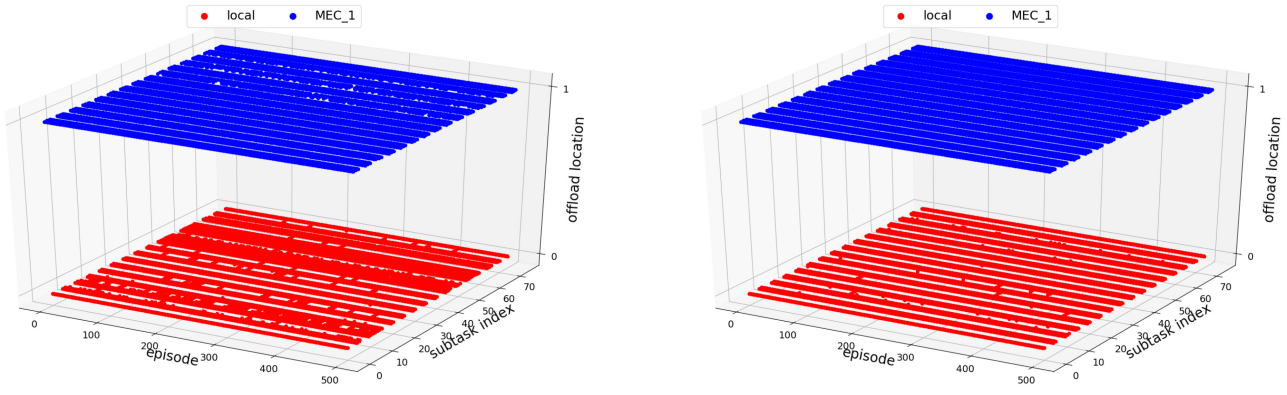

Fig. 16. Offloading location of different network states in single-MEC system (MEC = 25 GHz). (a) Poor wireless network state. (b) Good wireless network state.

the offloading location of the wireless network status in poor and good, respectively, when own MEC computing resource is 25 GHz. The wireless network state of the user is good when the uplink/downlink transmission rate is larger than 20 Mb/s. The wireless network state of the user is poor when the uplink/downlink transmission rate is lower than 10 Mb/s. There are more irregular points in Fig. 16(a) than in Fig. 16(b). A few irregular points in Fig. 16(b) with good network state are caused by network training errors.

Fig. 15(a) shows the offloading location of subtasks in the multi-MEC system when the computing resources of own MEC server are 5 GHz. Since the computing resources of own MEC server are insufficient, some of these subtasks (tracker, mapper, and object recognizer) are offloaded to adjacent MEC servers. When the computing resources of own MEC server are sufficient, most of these subtasks (tracker, mapper, and object recognizer) are offloaded to own MEC server. If subtasks are offloaded to adjacent MEC servers, the execution delay of the AR task will be increased. Therefore, subtasks do not need to be offloaded to adjacent MEC servers when the computing resources of own MEC server are sufficient. The offloading location of Fig. 14(b) is close to that of Fig. 15(b), and the explanations of Figs. 11(c) and 12 are verified.

VI. Conclusion

In this article, a joint optimization problem of resource allocation and task offloading for AR application in both the single-MEC system and multi-MEC system is formulated to minimize the energy consumption of each user subject to the latency requirement and limited resources. In order to conform to reality and make rational use of resources, we split an AR application into five subtasks according to their functions, and model it as a directed acyclic graph. To solve this problem, we propose a deep reinforcement learning algorithm based on the MADDPG framework. Each user is regarded as an agent competing for communication and computing resources. The optimized resource allocation and task offloading schemes are obtained by the well-trained MADDPG network. We provide numerical simulation results in terms of the convergence property, reward, and energy consumption, which shows that our proposed algorithm has better performance in terms of the energy consumption of users.

References

[1] G. Premsankar, M. Di Francesco, and T. Taleb, "Edge computing for the Internet of Things: A case study," *IEEE Internet Things J.*, vol. 5, no. 2, pp. 1275–1284, Apr. 2018.

[2] S. Wang *et al.*, "Augmented reality as a telemedicine platform for remote procedural training," *Sensors*, vol. 17, no. 10, p. 2294, 2017.

[3] N. Jamiat and N. F. N. Othman, "Effects of augmented reality mobile apps on early childhood education students' achievement," in *Proc. 3rd Int. Conf. Digit. Technol. Educ.*, 2019, pp. 30–33.

[4] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, "Mobile edge computing—A key technology towards 5G," ETSI, Sophia Antipolis, France, White Paper, pp. 1–16, 2015.

[5] "Mobile-edge computing," ETSI, Sophia Antipolis, France, White Paper, 2014.

[6] X. Huang, L. He, X. Chen, G. Liu, and F. Li, "A more refined mobile edge cache replacement scheme for adaptive video streaming with mutual cooperation in multi-mec servers," in *Proc. IEEE Int. Conf. Multimedia Expo (ICME)*, London, U.K., 2020, pp. 1–6.

[7] X. Huang and L. He, "Playback experience driven cross layer optimisation of app, transport and mac layer for video clients over long-term evolution system," *IET Commun.*, vol. 14, no. 13, pp. 2176–2188, Aug. 2020.

[8] C. Wang, C. Liang, F. R. Yu, Q. Chen, and L. Tang, "Computation offloading and resource allocation in wireless cellular networks with mobile edge computing," *IEEE Trans. Wireless Commun.*, vol. 16, no. 8, pp. 4924–4938, Aug. 2017.

[9] Y. Hao, M. Chen, L. Hu, M. S. Hossain, and A. Ghoneim, "Energy efficient task caching and offloading for mobile edge computing," *IEEE Access*, vol. 6, pp. 11365–11373, 2018.

[10] M. Liu, F. R. Yu, Y. Teng, V. C. M. Leung, and M. Song, "Computation offloading and content caching in wireless blockchain networks with mobile edge computing," *IEEE Trans. Veh. Technol.*, vol. 67, no. 11, pp. 11008–11021, Nov. 2018.

[11] S. Bi and Y. J. Zhang, "Computation rate maximization for wireless powered mobile-edge computing with binary computation offloading," *IEEE Trans. Wireless Commun.*, vol. 17, no. 6, pp. 4177–4190, Jun. 2018.

[12] Y. Wu, K. Ni, C. Zhang, L. P. Qian, and D. H. K. Tsang, "NOMA-assisted multi-access mobile edge computing: A joint optimization of computation offloading and time allocation," *IEEE Trans. Veh. Technol.*, vol. 67, no. 12, pp. 12244–12258, Dec. 2018.

[13] T. X. Tran and D. Pompili, "Joint task offloading and resource allocation for multi-server mobile-edge computing networks," *IEEE Trans. Veh. Technol.*, vol. 68, no. 1, pp. 856–868, Jan. 2019.

[14] L. Yang, H. Zhang, M. Li, J. Guo, and H. Ji, "Mobile edge computing empowered energy efficient task offloading in 5G," *IEEE Trans. Veh. Technol.*, vol. 67, no. 7, pp. 6398–6409, Jul. 2018.

[15] X. Chen and G. Liu, "Joint optimization of task offloading and resource allocation via deep reinforcement learning for augmented reality in mobile edge network," in *Proc. IEEE Int. Conf. Edge Comput.*, 2020, pp. 76–82.
[16] *IBM and Nokia Siemens Networks Announce World's First Mobile Edge Computing Platform*, IBM, Armonk, NY, USA, 2013. [Online]. Available: http://www-03.ibm.com/press/us/ en/pressrelease/40490.wss
[17] "Mobile-edge computing—Introductory technical white paper," ETSI, Sophia Antipolis, France, White Paper, pp. 1–36, 2014.
[18] T. X. Tran, A. Hajisami, P. Pandey, and D. Pompili, "Collaborative mobile edge computing in 5G networks: New paradigms, scenarios, and challenges," *IEEE Commun. Mag.*, vol. 55, no. 4, pp. 54–61, Apr. 2017.
[19] R. Al-Zaidi, J. Woods, M. Al-Khalidi, K. M. A. Alheeti, and K. McDonald-Maier, "Next generation marine data networks in an IoT environment," in *Proc. IEEE 2nd Int. Conf. Fog Mobile Edge Comput. (FMEC)*, Valencia, Spain, 2017, pp. 50–55.
[20] J. Zhang, W. Xia, F. Yan, and L. Shen, "Joint computation offloading and resource allocation optimization in heterogeneous networks with mobile edge computing," *IEEE Access*, vol. 6, pp. 19324–19337, 2018.
[21] I. Ketykó, L. Kecskés, C. Nemes, and L. Farkas, "Multi-user computation offloading as multiple knapsack problem for 5G mobile edge computing," in *Proc. IEEE Eur. Conf. Netw. Commun. (EuCNC)*, Athens, Greece, 2016, pp. 225–229.
[22] H. Zhao, Y. Wang, and R. Sun, "Task proactive caching based computation offloading and resource allocation in mobile-edge computing systems," in *Proc. IEEE 14th Int. Wireless Commun. Mobile Comput. Conf. (IWCMC)*, Limassol, Cyprus, 2018, pp. 232–237.
[23] J. Zhang *et al.*, "Energy-latency tradeoff for energy-aware offloading in mobile edge computing networks," *IEEE Internet Things J.*, vol. 5, no. 4, pp. 2633–2645, Aug. 2018.
[24] T. Q. Dinh, J. Tang, Q. D. La, and T. Q. S. Quek, "Offloading in mobile edge computing: Task allocation and computational frequency scaling," *IEEE Trans. Commun.*, vol. 65, no. 8, pp. 3571–3584, Aug. 2017.
[25] Q. Liu, S. Huang, J. Opadere, and T. Han, "An edge network orchestrator for mobile augmented reality," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, Honolulu, HI, USA, 2018, pp. 756–764.
[26] Y. Sun, Z. Chen, M. Tao, and H. Liu, "Communication, computing and caching for mobile VR delivery: Modeling and trade-off," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Kansas City, MO, USA, 2018, pp. 1–6.
[27] A. Akter, A. Islam, and S. Y. Shin, "Mobile edge computing based mixed reality application for the assistance of blind and visually impaired people," in *Proc. 7th Int. Conf. Inf. Commun. Technol. (ICoICT)*, Kuala Lumpur, Malaysia, 2019, pp. 1–5.
[28] R. Chai, J. Lin, M. Chen, and Q. Chen, "Task execution cost minimization-based joint computation offloading and resource allocation for cellular D2D MEC systems," *IEEE Syst. J.*, vol. 13, no. 4, pp. 4110–4121, Dec. 2019.
[29] Y. Wang, M. Sheng, X. Wang, L. Wang, and J. Li, "Mobile-edge computing: Partial computation offloading using dynamic voltage scaling," *IEEE Trans. Commun.*, vol. 64, no. 10, pp. 4268–4282, Oct. 2016.
[30] X. Yang *et al.*, "Communication-constrained mobile edge computing systems for wireless virtual reality: Scheduling and tradeoff," *IEEE Access*, vol. 6, pp. 16665–16677, 2018.
[31] J. Liu and Q. Zhang, "Offloading schemes in mobile edge computing for ultra-reliable low latency communications," *IEEE Access*, vol. 6, pp. 12825–12837, 2018.
[32] J. Hu, M. Jiang, Q. Zhang, Q. Li, and J. Qin, "Joint optimization of uav position, time slot allocation, and computation task partition in multiuser aerial mobile-edge computing systems," *IEEE Trans. Veh. Technol.*, vol. 68, no. 7, pp. 7231–7235, Jul. 2019.
[33] T. Verbelen, P. Simoens, F. De Turck, and B. Dhoedt, "Leveraging cloudlets for immersive collaborative applications," *IEEE Pervasive Comput.*, vol. 12, no. 4, pp. 30–38, Oct.–Dec. 2013.
[34] W. Zhan, C. Luo, J. Wang, G. Min, and H. Duan, "Deep reinforcement learning-based computation offloading in vehicular edge computing," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, 2020, pp. 1–6.
[35] S. Yu, R. Langar, X. Fu, L. Wang, and Z. Han, "Computation offloading with data caching enhancement for mobile edge computing," *IEEE Trans. Veh. Technol.*, vol. 67, no. 11, pp. 11098–11112, Nov. 2018.
[36] Q. Li, S. Wang, A. Zhou, X. Ma, F. Yang, and A. X. Liu, "QoS driven task offloading with statistical guarantee in mobile edge computing," *IEEE Trans. Mobile Comput.*, early access, Jun. 23, 2020, doi: 10.1109/TMC.2020.3004225.
[37] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," in *Advances in Neural Information Processing Systems*. Red Hook, NY, USA: Curran, 2017, pp. 6379–6390.
[38] B. Gu, X. Yang, Z. Lin, W. Hu, M. Alazab, and R. Kharel, "Multi-agent actor-critic network-based incentive mechanism for mobile crowdsensing in industrial systems," *IEEE Trans. Ind. Informat.*, early access, Sep. 21, 2020, doi: 10.1109/TII.2020.3024611.
[39] Y. Zhang, Z. Mou, F. Gao, J. Jiang, R. Ding, and Z. Han, "UAV-enabled secure communications by multi-agent deep reinforcement learning," *IEEE Trans. Veh. Technol.*, vol. 69, no. 10, pp. 11599–11611, Oct. 2020.
[40] T. P. Lillicrap *et al.*, "Continuous control with deep reinforcement learning," 2015. [Online]. Available: arXiv:1509.02971.
[41] J. Zheng, Y. Cai, Y. Wu, and X. Shen, "Dynamic computation offloading for mobile cloud computing: A stochastic game-theoretic approach," *IEEE Trans. Mobile Comput.*, vol. 18, no. 4, pp. 771–786, Apr. 2019.
[42] X. Ma, C. Lin, H. Zhang, and J. Liu, "Energy-aware computation offloading of IoT sensors in cloudlet-based mobile edge computing," *Sensors*, vol. 18, no. 6, p. 1945, 2018.
[43] X. Long, J. Wu, and L. Chen, "Energy-efficient offloading in mobile edge computing with edge-cloud collaboration," in *Proc. Int. Conf. Algorithms Archit. Parallel Process.*, 2018, pp. 460–475.

Xing Chen received the M.S. degree in information and communication engineering from Lanzhou University of Technology, Lanzhou, China, in 2017. He is currently pursuing the Ph.D. degree in information and communication engineering with Xi'an Jiaotong University, Xi'an, China.

His research focus includes mobile-edge computing and deep reinforcement learning.

Guizhong Liu (Member, IEEE) received the B.S. and M.S. degrees in computational mathematics from Xi'an Jiaotong University, Xi'an, China, in 1982 and 1985, respectively, and the Ph.D. degree in mathematics and computing science from Eindhoven University of Technology, Eindhoven, The Netherlands, in 1989.

He is a Full Professor with the School of Electronics and Information Engineering, Xi'an Jiaotong University. His research interests include nonstationary signal analysis and processing, image processing, audio and video compression, and inversion problems.